AD No. 735320

Reproduced by
NATIONAL TECHNICAL
INFORMATION SERVICE
Springfield, Va. 22151

UNCLASSIFIED

Security Classification

DOCUMENT CONTROL DATA - R&D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY *(Corporate author)* Lincoln Laboratory, M.I.T.	2a. REPORT SECURITY CLASSIFICATION Unclassified 2b. GROUP None
3. REPORT TITLE Graphics	
4. DESCRIPTIVE NOTES *(Type of report and inclusive dates)* Semiannual Technical Summary, 1 June 1971 through 30 November 1971	
5. AUTHOR(S) *(Last name, first name, initial)* Forgie, James W.	

6. REPORT DATE 30 November 1971	7a. TOTAL NO. OF PAGES 24	7b. NO. OF REFS None

8a. CONTRACT OR GRANT NO. F19628-70-C-0230	9a. ORIGINATOR'S REPORT NUMBER(S) Semiannual Technical Summary, 30 November 1971
b. PROJECT NO. ARPA Order 691	9b. OTHER REPORT NO(S) *(Any other numbers that may be assigned this report)* ESD-TR-71-305
c. d.	
10. AVAILABILITY/LIMITATION NOTICES Approved for public release; distribution unlimited.	
11. SUPPLEMENTARY NOTES None	12. SPONSORING MILITARY ACTIVITY Advanced Research Projects Agency, Department of Defense

13. ABSTRACT

With the exception of the tablet multiplexer and interprocessor interrupt boards, all Laboratory-built hardware for the Terminal Support Processor (TSP) system has been successfully integrated. While the manufacturer has yet to install the last of a series of modifications to the Meta 4 system, hardware performance and reliability appear adequate to support active software development and system checkout. Simple test programs have been sent from TX-2 to the TSP via the ARPA network and operated satisfactorily. A design change in the LIL language has replaced the implicit data segment with a return stack as a means of providing local storage for subroutines. The necessary changes have been incorporated in the firmware. A strategy for TSP main and disk memory management has been worked out which utilizes a feature of the disk controller design to effect garbage collection while swapping user programs.

Network software for the IBM 360/67 was certified as conforming to network standards on 20 July 1971. The corresponding software for TX-2 is approaching operational status. An experiment was carried out to explore the effects on speech transmission of the use of a message-switched communication system with characteristics similar to the ARPA computer network. Results indicated that speech could be satisfactorily handled by such a system.

Designs have been worked out for the TX-2 Speech Data Base and a sub-operating system called the Speech Processing Controller. The Speech Data Base is intended to provide fast, automatic access to the entire range of data associated with each of the many utterances to be dealt with in the development of analysis and recognition algorithms required for the phonetic recognition area of research on the use of speech for man-machine communication. The requirements which led to the data base and controller designs are discussed, and the systems are described at a conceptual level.

The BCPL compiler for TX-2 has been extended to handle previously discussed structure definitions, and a symbolic debugging system for BCPL programs is being developed.

A scan converter display system on TX-2 shows promise for the presentation of speech spectrograms and other gray level displays.

14. KEY WORDS

man-machine communications	computer graphics	ARPA Network
speech understanding systems	Terminal Support Processor (TSP)	TX-2 computer

UNCLASSIFIED

Security Classification

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
LINCOLN LABORATORY

GRAPHICS

SEMIANNUAL TECHNICAL SUMMARY REPORT
TO THE
ADVANCED RESEARCH PROJECTS AGENCY

1 JUNE 1971 – 30 NOVEMBER 1971

ISSUED 21 DECEMBER 1971

Approved for public release; distribution unlimited.

LEXINGTON MASSACHUSETTS

The work reported in this document was performed at Lincoln Laboratory, a center for research operated by Massachusetts Institute of Technology. This work was sponsored by the Advanced Research Projects Agency of the Department of Defense under Air Force Contract F19628-70-C-0230 (ARPA Order 691).

This report may be reproduced to satisfy needs of U.S. Government agencies.

Non-Lincoln Recipients

PLEASE DO NOT RETURN

Permission is given to destroy this document when it is no longer needed.

GRAPHICS

The ARPA-supported work in man-machine communications reported in the series of Semiannual Technical Summaries entitled "Graphics" has been reoriented toward work on speech understanding systems. This will be the last report in the "Graphics" series. Future reports will be entitled "Speech" but will continue to report on the status of the Terminal Support Processor system and other work in the computer graphics area.

SUMMARY

With the exception of the tablet multiplexer and interprocessor interrupt boards, all Laboratory-built hardware for the Terminal Support Processor (TSP) system has been successfully integrated. While the manufacturer has yet to install the last of a series of modifications to the Meta 4 system, hardware performance and reliability appear adequate to support active software development and system checkout. Simple test programs have been sent from TX-2 to the TSP via the ARPA network and operated satisfactorily. A design change in the LII language has replaced the implicit data segment with a return stack as a means of providing local storage for subroutines. The necessary changes have been incorporated in the firmware. A strategy for TSP main and disk memory management has been worked out which utilizes a feature of the disk controller design to effect garbage collection while swapping user programs.

Network software for the IBM 360/67 was certified as conforming to network standards on 20 July 1971. The corresponding software for TX-2 is approaching operational status. An experiment was carried out to explore the effects on speech transmission of the use of a message-switched communication system with characteristics similar to the ARPA computer network. Results indicated that speech could be satisfactorily handled by such a system.

Designs have been worked out for the TX-2 Speech Data Base and a sub-operating system called the Speech Processing Controller. The Speech Data Base is intended to provide fast, automatic access to the entire range of data associated with each of the many utterances to be dealt with in the development of analysis and recognition algorithms required for the phonetic recognition area of research on the use of speech for man-machine communication. The requirements which led to the data base and controller designs are discussed, and the systems are described at a conceptual level.

The BCPL compiler for TX-2 has been extended to handle previously discussed structure definitions, and a symbolic debugging system for BCPL programs is being developed.

A scan converter display system on TX-2 shows promise for the presentation of speech spectrograms and other gray level displays.

Accepted for the Air Force
Joseph R. Waterman, Lt. Col., USAF
Chief, Lincoln Laboratory Project Office

CONTENTS

GLOSSARY

APEX	The TX-2 time-sharing system
BCOM	BCPL Compilation Oriented Machine – the simulated machine for executing TSP software
BCPL	Basic Combined Programming Language – an intermediate-level language for computer programming
FDP	Fast Digital Processor – a Lincoln Laboratory computer designed for waveform processing applications
LIL	Local Interaction Language – a language for use in the TSP system
Meta 4	A microprocessor manufactured by Digital Scientific Corporation and utilized in the TSP system
NCP	Network Control Program
TELNET	The software which allows a console on one network computer to function as the console for another
TSP	Terminal Support Processor

GRAPHICS

I. TERMINAL SUPPORT PROCESSOR (TSP) SYSTEM

The TSP system is a small-scale computer system intended to support interactive graphics users of a computer network. The design aims at providing basic interactive graphics services for a number of consoles, each consisting of a keyboard, a tablet, and a pair of storage scopes. The system provides a language called LIL, which a user can utilize to control interactions between his console input and output devices and between the TSP and other computers in the network. The TSP itself consists of three microprocessors sharing 65k words of 900-nsec core memory arranged in 8 banks of 8k words each. Previous reports in this series have described the user specifications for LIL (31 May 1970, DDC AD-709187) and the system architecture of the TSP (30 November 1970, DDC AD-716817).

The following sections contain brief reports on the current status of the hardware and software components of the TSP system.

A. TSP Hardware

Previously reported difficulties which led to substantial delays in the availability of the Meta 4 dual-processor system have been largely overcome. Access to the system has been adequate to allow the connection and testing of all the Laboratory-built input-output interface hardware with the exception of the tablet multiplexer and the interprocessor interrupts. The tablet multiplexer interface design will be released to construction in December. The interprocessor interrupt hardware is ready for systems test.

After several months of satisfactory operation while connected to the SEL test computer, the LX-1 microprocessor and display generator have now been integrated into the TSP system, and display generating programs have been successfully executed from the Meta 4 core memory.

The manufacturer has not completed installation of the modifications to the Meta 4 system. However, current hardware performance and reliability appear adequate to support active software development.

B. TSP System Checkout

Since TX-2 is the source of all TSP software, the first IO connection to be tested was the interface to the ARPA network. This interface was tested by using TX-2 to generate test data and sending these data through the network to the Meta 4. The Meta 4 then sent the data back over the network to TX-2, which checked the echoed data against the data that it originally generated. Several million words were echoed in this manner. The data rates observed were low enough to give us confidence that we can load the Meta 4 core memory reliably from TX-2 through this path.

Programs have been written for TX-2 and the Meta 4 to load the Meta 4 core memory from TX-2 files using the network. Thus, we can use the TX-2 file system to store copies of TSP system programs. Such files can also be used as input to the Meta 4 simulator which runs on the TX-2. Debugging can therefore be carried out in either or both environments, the choice depending upon the nature of the trouble being investigated.

The Meta 4 program which receives these files and loads the Meta 4 core memory is fairly long, since it must deal with at least a portion of the network protocol (to discriminate against messages from other hosts, for example). Eventually, this program will be added to the microprogram memory of the Meta 4, from which it will be invoked by the push of a button. For the present, the program has been copied onto the disk. It can be read in by a seven-instruction program, which must be keyed in from the panel switches.

The above bootstrapping technique has been used to transfer a number of programs from the TX-2 to the Meta 4. For example, a program to transfer data from the Meta 4 to the LX-1 control memory, along with the LX-1 control memory data, was sent in this way. (The LX-1, the third microprocessor in the TSP system, has a writable control memory.) The program ran on the Meta 4 and loaded LX-1 control memory. This LX-1 program ran successfully, displaying the contents of the Meta 4 core memory. Thus, the connection between the Meta 4 and the LX-1 has been given a preliminary checkout.

The low-speed multiplexer for keyboards and indicators, described in the previous Semiannual Technical Summary, was also connected and tested.

A test program was run to determine the effect of using the disk to access the same memory bank in which a processor is executing read-alter-rewrite memory cycles. Since the disk requires every other memory cycle while data are being transferred, the experiment was designed to determine the utility of read-alter-rewrite cycles in the system. We concluded that read-alter-rewrite cycles could be used successfully if the microcode was written with care to avoid unnecessarily long lockout of the memory bank.

A comprehensive set of central processor diagnostics has been run. A few cases have occurred in which our interpretation of the microcode specification was incorrect. Fortunately, we will be able to eliminate these trouble spots with only minor changes in the microcode.

Most of the system programs for the TSP will be written in BCPL. The TX-2 BCPL compiler has been modified to generate code for the BCOM machine. A compatible assembler has been written to handle those parts of the code which must be written in assembly language. Separately compiled and assembled segments may be combined with the standard TX-2 loader, and a final utility program puts the loader output into the proper format to be sent to the Meta 4. Our first attempt to run BCPL code on the Meta 4 showed a minor bug in the assembler. No serious problems in correcting the difficulty are expected.

C. TSP Software

1. LIL

Considerations arising in the design of the TSP core management software have led to some design changes in LIL. The principal change with respect to the user specifications reported previously is the elimination of the implicit data segment related to a program segment. In the new design, subroutine return information and local storage for a subroutine are kept in a return stack which is referenced relative to base register zero. A subroutine may access only that portion of the stack associated with its invocation. The design does not require that portions of the stack occupy contiguous core locations, and those portions which are not currently active will be kept on the disk. The new design allows subroutines to be called recursively and should result in more efficient core and disk usage.

Other changes in the LIL implementation plans affect the handling of scratch memory which contains the LIL general registers, base registers, the active segment table, and other key information. The structure of the program segment has been modified to allow for a more efficient search of the segment for individual subroutines.

In order to carry out these changes, substantial portions of the LIL firmware (Meta 4 microcode) have been rewritten and checked out on the Meta 4 simulator.

2. LILA

The LIL assembler (LILA) is now operational and is being used to generate test programs for the LIL firmware. LILA was written in BCPL, taking care to avoid machine-dependent features of BCPL. LILA can therefore be transferred easily to the 360/67 or any other machine with a BCPL compiler.

3. TSP Core and Disk-Management

A plan for the management of TSP main memory and disk has been worked out. In this design the disk will be divided into four areas. Area 1 will contain the TSP resident system. Area 2 will consist of 6000 words for each console as a fast swap area to hold each console's active segments when they must be swapped out. Area 3 will be used to buffer data going to or coming from the various IO devices. Area 4 will be a bulk storage area where inactive program and data segments will be kept.

The TSP main memory consists of eight modules of 8192 words each. The first three or four modules will contain the TSP system. At least two (three, if space permits) modules will be assigned to user's programs. The remaining space will be used for IO buffers.

The two or three modules assigned to user's programs will each be further divided into three areas. Area 1 (4096 words) will contain the user's active segments. Area 2 (2048 words) will contain system information such as scratch memory, active segment table, return stack, etc. The remaining words in these modules will be utilized for purposes to be selected to minimize memory conflict problems. Since the disk rate is very high (the disk takes every other memory cycle), there is a reduction in the effective speed of the remaining memory in the modules in which disk swapping occurs regularly. Depending on whether two or three modules are available for users, the programs of two or three users can be in core simultaneously. A user's program will be set up and running in one module while other user's programs are being swapped in and out of the other module(s).

A user is allowed to add or delete items and subroutines from his program segments dynamically. When he deletes something in a segment, a bit is set. Garbage collection is accomplished when he is next swapped out. The TSP disk controller allows data to be gathered from scattered blocks in memory and written on the disk as one continuous record. When such a continuous record is read back into core, the only delay encountered is the rotation time required to reach the beginning of the record. Pointers in the item and subroutine headers of the program segments allow the use of this disk controller feature to accomplish garbage collection on the way to the disk. Transfer in both directions will proceed at full disk speeds.

The user scheduling algorithm will follow an aggressive swapping policy. Everything except information about the user's IO streams will be written to the disk whenever (1) the user exhausts his allotted time interval, (2) he requires new information from the disk, or (3) garbage collection is required to provide working space. Segments which have become inactive will be written to the bulk storage area of the disk with garbage collection within the segment taking place if required. Disk operations in the bulk area will be less efficient, since less control of latency losses is possible.

II. ARPA NETWORK

A. 360/67 Network Software

The Network Control Program (NCP), LOGGER, and TELNET programs for the IBM 360/67 system were modified to conform with revised network protocol requirements, and performance of the modified programs was certified by the network group at RAND Corporation on 20 July 1971. Subsequent use of the network facilities has uncovered some deficiencies in the design and implementation. Most evident is the inability of a remote user to access the 360/67 from a terminal IMP (TIP) which poses more stringent protocol demands than those required for certification. As a result, the system does not appear as fully operational in the periodic testing of the status of network host computers carried out by Bolt Beranek and Newman (BBN) from a TIP terminal. This and other problems are currently under investigation, together with a study of the performance and data transfer rates through the NCP.

B. TX-2 Network Software

The basic network support software, the NCP, TELNET, and LOGGER programs for TX-2, are approaching operational status. Basic NCP functions are being performed satisfactorily, and reliability is approaching an acceptable level as operating experience grows with the checkout of TELNET and LOGGER programs.

The TELNET USER program is the software which allows a local console to be used as a console for any network computer. An early version of the TX-2 TELNET USER program has been successfully used on several occasions to log in to the PDP-10 TENEX system at BBN and to communicate with several programs there in a satisfactory manner. In order to make such two-way communication practical, a reasonably convenient transformation between the TX-2 character code set and ASCII, the network standard for TELNET usage, has been devised. Further work is indicated to make usage more convenient and to allow for the various eventualities that may occur while logged in on another system.

The TELNET SERVER and LOGGER programs allow use of the local host computer from other points in the network. While the TX-2 TELNET USER program is implemented as user level software, the TELNET SERVER and LOGGER require a combination of user level and supervisor routines. The necessary changes in the APEX time-sharing supervisor have been carried out, and a preliminary version of the user level programs is nearly operational. The user level TELNET SERVER programs operate as interrupt driven programs in the user's virtual machine in such a way as to make their actions completely transparent to other user level software. As in the case of the TELNET USER program, further work is indicated to handle in a more satisfactory fashion the many special situations which can occur in network usage.

C. Speech Transmission in a Message-Switched Network

Some experiments have been carried out to explore the effects on speech communication of a message-switched communication system as opposed to the familiar channel-switched system such as the telephone system. In the channel-switched system, the user may experience delays in obtaining a connection to the party with whom he wishes to communicate, but once such a connection is established, his speech passes through the system with a uniform delay which is generally quite short with respect to the pacing of speech events. In a message-switched system such as the ARPA computer network, a channel always exists between parties, and messages are multiplexed over this channel. With this type of system there is no consequential delay associated with establishing a connection, but the individual utterances of a speaker are subjected to delays which vary according to other traffic in the system and are likely to be significant with respect to the pacing of speech events. Such delays, if not properly handled, could be expected to affect the acceptability, if not the intelligibility, of a speech communication system of this kind. The experiments that have been made with TX-2 and the ARPA network allowed the exploration of the effectiveness of a class of algorithms in minimizing the damaging effects of message delays on conversational speech.

The ARPA network is currently interconnected with 50-kilobit telephone lines. Such a bandwidth is not sufficient for multiplexing digitized speech which requires the order of 50 kilobits for a single conversation. However, the use of a vocoder can reduce the bandwidth sufficiently to allow multiplexing a reasonable number of conversations. For example, vocoders are available using bandwidths of 2400 bits per second and producing speech of acceptable quality. Since there is no need to transmit anything during the silent portions of a conversation, one could assume something like 1200 bits per second as a nominal one-way message load placed on the network by a person carrying on a conversation.

For the experiment, a pair of telephone handsets was connected to TX-2 through analog-to-digital and digitial-to-analog converters so that two persons could converse in a full-duplex fashion with both speech signals passing through TX-2 core memory in digital form. Programs monitored the amplitude of the signal from each telephone transmitter and stored the signal in memory only when the amplitude was above a threshold. Periods of time when the input signal was below the threshold were considered to be silent, and the signal sent to the telephone receiver for such periods was either obtained from a block of memory containing all zeros ("true silence") or was obtained by repeating any arbitrary section of the speech signal according to the algorithm being explored.

For the purpose of introducing network delays, messages were made up corresponding to blocks of the speech input. The messages were transmitted to "fake host 3" at some IMP site in the network. "Fake host 3" is a feature of the network implementation which allows experiments such as this to be carried out without involving personnel at remote sites. A message received by "fake host 3" is acknowledged and discarded. In our experiment, the receipt of the acknowledgment at TX-2 was used as an indication that the block of data corresponding to the acknowledged message could be reconstituted as speech at any time thereafter. By choosing IMP's at different locations around the country as recipients for the messages, it was possible to achieve a range of message propagation delays, and by varying the size of the message corresponding to a block of speech, the effect of message load could be explored. The choice of message size corresponds to the assumption of a vocoder with a particular degree of bandwidth compression. Actual values used in experiments ranged from about 2400 bits per second of speech to about 16,000 bits per second. The use of a real vocoder in the input/output path would have yielded a more realistic experiment by introducing the small loss in intelligibility and reduced naturalness associated with vocoded speech, but resources were not available to interface the vocoder hardware, and TX-2 is not fast enough to simulate a vocoder in real time.

In a message-switched speech communication system, the crucial problem at the transmitting end is to select a block size for transmission. If too short a block is selected, message efficiency will be unnecessarily low because the overhead for routing and message acknowledgment is independent of message length. If too long a block size is chosen, over-all message delay will be excessive. In our experiments, block sizes were limited to multiples of 10 milliseconds, and the actual block length was an experimental variable. Results showed that the optimum size should depend upon the distance (more properly, the number of intervening IMP's) between source and destination. Block sizes of 80-msec duration were usable for close IMP's, but 280 msec appeared optimum for transmission to the West Coast. Of course, if the input speech fell below threshold before the end of a nominal block, the actual block was terminated and an indication of the beginning of a silent interval was transmitted along with the block.

At the receiving end, the crucial decision is when to begin reconstituting the speech. If the receiver starts too soon after receipt of the first block following a silent interval, the receiver may finish reconstituting all received blocks before the message carrying the next portion of the utterance has arrived. In that event, some sort of "glitch" will be introduced in the output speech. In our experiment, the method of handling a glitch was to repeat the most recent 10 msec of output for a time and then switch to "true silence" until the next block of speech arrived. This algorithm was adopted on the argument that brief silences (the order of 20 to 30 msec) could be more disturbing in the middle of a continuous utterance than the stretching effect of repeating the last portion of speech. The adequacy of this algorithm was not thoroughly investigated in the experiment, but it appeared to give reasonable results and could be expected to be quite satisfactory if a true vocoder were used, in which case the repetition would involve the last vocoder frame rather than the last 10 msec of speech which sometimes produces an undesirable false 100-Hz pitch sensation.

To avoid glitches in the reconstituted speech, the receiver could delay reconstitution until all the blocks between silent intervals have been received. Any variation in network delays could then be taken up in the duration of the silent portions of a conversation and would for the most part pass unnoticed. Unfortunately, such a procedure would result in excessive over-all delays (several seconds) and excessive buffer storage at the receiver. A practical system must risk occasional glitches and begin reconstituting after a more modest delay. We do not have sufficient data to adequately evaluate the trade-off between glitch probabilities and delays or to assess the relative degree of disturbance to a human user caused by glitches and delays, but our experiments indicated that satisfactory settings of the delay and block length parameters could be found for all the network conditions that were explored. For example, transmissions to the West Coast (our worst case) were satisfactory with a block size greater than 220 msec and delays on the order of 500 msec. Of course, reconstitution was always permitted to begin without delay if a received block string ended in a silent interval.

Adequate evaluation of speech communication systems requires extensive testing with many talkers and listeners to obtain even relative quality judgments. No such testing has been carried out for the experimental situation described here. While we have quantitative data on network delays and measurements of the traffic resulting from our experiments, we have no data on other network activity at the time when our experiments were being conducted. Our principal results are therefore the subjective evaluations made by the small number of people who have used the experimental system or listened to recordings of conversations conducted through the system. The listeners have generally agreed that communication can be quite satisfactory when parameters are properly adjusted and very bad when they are incorrectly set. The main noticeable effect with proper settings is the tendency to lose the first and last sounds of an utterance, a characteristic behavior of a voice-operated switch which is to be expected from the action of our silence threshold measurement. This effect could be reduced by a more complex threshold algorithm, but no attempt was made to improve performance in this regard. With parameter settings which produce occasional glitches, the listener is aware that something unusual is happening to the speech he is hearing, but he has no trouble following the conversation. With more adverse parameter settings, the reconstituted speech begins to fall progressively further behind the input speech and causes an effect that has been called "tired," "inebriated," or "senile." With extreme settings, the output becomes so chopped up as to be unintelligible. Such pathological behavior is actually rather difficult to achieve and requires parameter settings far from any reasonable operating setting in a real system. The other principal effect to be observed is the average delay between $\frac{1}{2}$ and 1 second associated with glitch-free operation. Our users found this magnitude of delay to be definitely noticeable, particularly when trying to interrupt the speaker at the other end of the line, but they agreed that they could readily adapt to the delay and did not find it objectionable.

Our conclusion from these experiments is that speech communication in a message-switched communication system with delay characteristics similar to a lightly loaded ARPA computer network could be quite satisfactory from a human-factors standpoint.

III. SPEECH

Speech oriented work has concentrated on the design and implementation of a software base for supporting research in phonetic recognition on TX-2 and the Laboratory's Fast Digital Processor (FDP). In the over-all plan, the FDP will handle the bulk of the speech processing, computing spectra, tracking formants, and making feature measurements. TX-2 will handle complex recognition logic and higher level processing as well as programs to support performance evaluation. Short-term goals for the FDP are the realization of programs for basic speech processing, and for TX-2, the achievement of data base support software and interactive control and labeling programs. Programs to compute homomorphic spectra and track formants are currently operating on the FDP, but discussion of them will be deferred until the next report in this series when more data on their performance should be available. The remainder of this section will be devoted to a presentation of the designs for the TX-2 Speech Data Base and for a suboperating system called the Speech Processing Controller intended to provide the programmer with tools to build large program structures from individual modules.

A. Speech Data Base

The Speech Data Base is intended to provide fast, automatic access to the entire range of data associated with each of many utterances. Flexible retrieval based on characteristics of the speech signal, as well as phonetic and lexical content, is to be provided. The detailed form of the Speech Data Base, which may also serve the "front-end" processing needs of other ARPA-sponsored speech projects, has been worked out. Programs are at present being specified.

1. Functional Requirements

The design of any substantial system represents a compromise among many competing requirements. The following list of requirements represents those which have been given active consideration in the design of the data base.

(a) The data base will be maintained and accessed in the environment of the APEX time-sharing system. It will share the same file storage device. Therefore, the data base services must be coordinated with normal APEX user service activities, such as swapping and file access.

(b) Users logged in to APEX will have what appears to them to be concurrent read and write access to the data base.

(c) Protection against effects of hardware and software crashes, write protection (i.e., ownership), and protection against data being changed while reading it will be provided, but all data will be available to all users.

(d) An effectively open-ended number of data entities or other attributes may be associated with an utterance. Data entities will differ in size from waveform data at about 100,000 bits per second of speech to an array that records the occurrence of three or four acoustic events in an utterance.

(e) Relations among attributes such as speaker, nominal lexical content, etc., and utterances that possess such attributes must be maintained.

(f) "Sub-data bases," i.e., lists of occurrences of classified acoustic, phonetic, morphological, etc., events, must be constructable.

(g) List of attributes that would be common to a number of utterances, such as nominal lexical transcriptions, must be maintained.

(h) Combinations of the above two types of lists must be possible, effectively enabling the user to produce an inverted file structure.

(i) Users must be able to record comments freely at the level of utterances or data entities.

(j) There must be adequate descriptor space associated with data entities. This includes conventions about the basic unit of classification, data type, etc.

(k) There must be facilities for descriptor-oriented retrieval of data entities or attributes, as well as context oriented retrieval, e.g., retrieval of all examples of unvoiced stops preceding front vowels. There must be means for interfacing user-provided programs with the basic retrieval routines so that arbitrary content-directed retrieval is possible. For example, it should be possible to locate all instances where the second formant motion exceeds some rate, r, for a time, t.

(l) Time-oriented descriptors must have a special status. There must be means of associating time of occurrence with acoustic and other attributes within an utterance. Storage and retrieval functions may be time selective as well as data-type selective.

(m) ARPA network access to the data base must be reasonably convenient. Extending the data base to other network facilities such as the Datacomputer being developed by Computer Corporation of America should be straightforward.

It should be noted that current discussions with representatives of the ARPA-supported speech projects which may become users of the Lincoln data base may introduce new requirements or suggest a different balance in the design trade-offs.

2. Data Base Organization

The data base is organized as a simple three-level tree structure, shown in Fig. 1. A data entity such as a waveform or a spectrum, formant track, etc., is at the bottom of the structure. The node from which branches to such data entities fan out is an "utterance entry." The set of such entries (including a few other types of entries that will be mentioned below) is called the Catalog. For every entry in the Catalog, there is a pointer in the topmost part of the structure, which is called the Index.

a. Contents of Index

An entry in the Catalog is identified by its serial number. This serial number is derived from the serial position of its pointer in the Index. Along with the pointer to the entry, the Index keeps an indication of the entry's length. An entry type indication is also recorded.

b. Contents of Utterance Entry

The utterance entry keeps track of the locations and descriptions of the various data entities associated with an utterance. The form of an utterance entry is an effectively open-ended concatenation of variable length fields. All fields have the following basic contents:

(1) Length in TX-2 words.

(2) Type, identical to the data type of the thing that the field is keeping track of.

(3) Place where the data are to be found, i.e., on the TX-2 drum, magnetic tape, in the entry itself as immediate data, or somewhere else in the network.

(4) Owner, the user who can change or delete the field.

(5) Actual drum, tape or network address if not immediate data.

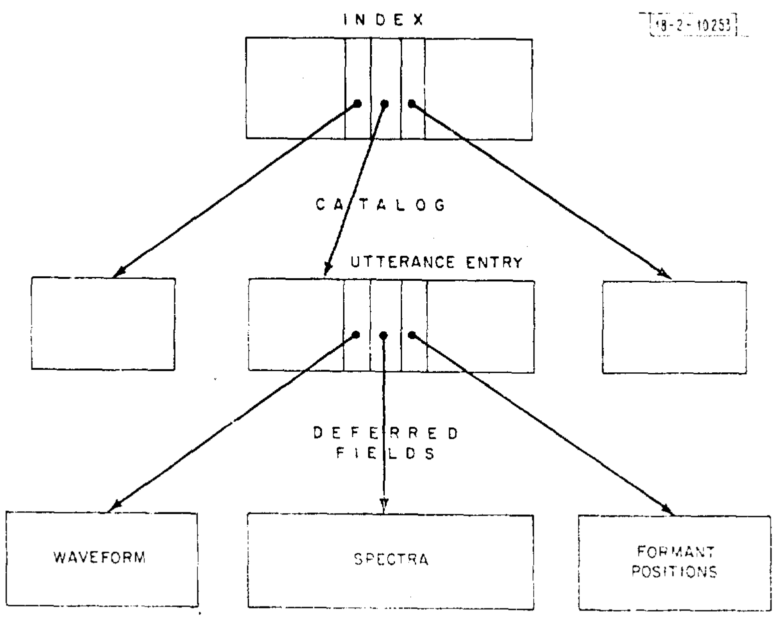


Fig. 1. Structure of Lincoln Speech Data Base.

Fields for data entities will contain appropriate descriptors. For example, fields for the various kinds of arrays will all have the following descriptors:

(1) Number of rows.

(2) Number of columns.

(3) Starting time of the time span covered by the data.

(4) Ending time of the time span covered by the data.

c. Data Entities

As indicated above, the data entity may be stored immediately within its field and hence in the utterance entry. It is expected that short data entities would be stored in this way. For longer data entities, the field contains a drum or tape location. The thing in which the data entity is stored when it is not with its field is called a "deferred field." Whichever place the data entity appears, it is preceded by the following descriptors:

(1) Utterance serial number of the "parent" utterance.

(2) Data type, same as the field i.d.

(3) Date and time at which the data entity was written.

d. Two Special Fields

A global descriptor field in an utterance entry contains several items of information. These include identifiers for the normal lexical and phonetic transcriptions, the speaker, and a descriptor for binary characteristics such as sex of the speaker.

The means of associating times in an utterance with acoustic or phonetic events is a data entity called a time-event array. For example, a time-event array will record the results of labeling the segments in an utterance. It is a two-dimensional array with a row for each segment. The first column contains the start time of the segment. The second column contains the end time of the segment. The third column contains the encoded phonetic classification of the segment.

e. Private Data Entities

An utterance entry keeps track of both "standard" entries, those that are expected to exist for all utterances, and data entities that are generated in the course of an individual's work. An individual may prefer not to recompute results on which various decision strategies are to be applied. He can and should store his intermediate results in the data base. The appropriate utterance entry will contain a field to keep track of his private data. His ownership will be recorded via the "owner" descriptor in the field.

It is expected that the user will frequently work with data on a time selective basis, i.e., storing and retrieving data and results for particular time spans. The storage and retrieval system has the capability to be time selective, that is, to keep track of start- and end-time descriptors, and to retrieve only those sections of data which are going to be looked at by the processing routine even though the selected portions of data were not stored as separately identified data entities. For high bit rate data such as waveforms or spectra, this capability can save considerable quantities of memory space and processing time in experiments involving the analysis of particular phonetic events.

f. Other Types of Catalog Entries: List Entries

There are three kinds of lists that appear to be necessary, and these will be kept at the level of entries in the Catalog. There are lists of attributes that are shared by a number of utterances, such as speaker, nominal transcriptions, etc.; lists of utterances or chunks of utterances that are examples of a particular attribute; and combination lists, which contain multiple attributes with each of which is associated a list of the utterances or chunks which exemplify it.

The list entries have a single owner. All data are immediate. The fields that contain the data are structured appropriately for the purpose they serve.

3. Implementation and Interfacing Considerations

The software to support the storage and retrieval (S&R) system is being designed as a combination of user level and APEX supervisor programs. APEX will be extended to assume responsibility for the management of secondary memory facilities and the protection of the data base against accidental damage. The user level portions of the S&R system will operate as an interface program between analysis or recognition programs and the basic supervisor functions. These programs will assume responsibility for all conventions regarding field and entry types and the internal format of information in the data base. These programs are being designed to interface uniformly to BCPL programs and SPC processes (described below).

B. Speech Processing Controller

The Speech Processing Controller (SPC) has been designed to provide a programming environment in which individual programs could be readily organized into a coherent system. To meet our needs for speech processing support, many subsystems must work together: facilities for data display, storage and retrieval, labeling, segmentation, etc. The needed programming environment is intended to organize communication among programs and subsystems, to enable individual programs and subsystems to be called conveniently, and to change the order and conditions of calling, and/or the computation of arguments, without going back to the compiler. As a controller, the SPC is intended to serve as a command interpreter for executing sequences of statements (processes) and for direct communication with a user.

In order to facilitate communication among programs and subsystems, the SPC establishes a "common" area for programs where global variables are kept. The entities in this common area can be referenced symbolically in the statements that are interpreted by the SPC. By making the common area accessible to the interpreter, the SPC greatly increases the ability of the programmer to control, communicate with, and monitor an individual program or a subsystem. This is one of several senses in which the SPC can be considered as an interpreter at the programmer's level.

One may note that most interpreters are intended to insulate the user from what happens at the level of the programs that are eventually called. The SPC, on the contrary, enables the user to get back to the level of statements in the programming language without going to the compiler, and this feature may be its most novel aspect.

Subroutines are declared to the SPC and so can be called by name in SPC statements. Since, except for "primitive" level programs, modules communicate arguments through the common area, and arguments can be supplied from the interpreter statements. Thus, the structure formed by a set of program modules can be developed and reordered in the SPC without going back to the programming language.

The user can input statements to the SPC for immediate execution via the keyboard. Since arguments associated with graphic interactions, such as character recognized, pen location, etc., can be maintained in the common area, the user can run processes as a result of his actions at the tablet. Thus, the SPC implements an effective interactive capability.

SPC statements have been designed to facilitate the job of controlling a subsystem or activity such as interactive labeling of speech segments. Statement syntax includes conditionals, statement labels, computed "go-to," dummy arguments, and automatic local storage. A non-local "go-to" has proved particularly useful in transferring control among modules that represent different system states in an interactive environment.

A major function of the SPC is the management of the programmer's data memory space. Since the 17-bit address length of TX-2 is not necessarily sufficient to allow all data items of interest to a program to be laid out in the addressable space, the programmer must exercise considerable care in dealing with applications such as speech processing which exceed the addressing limitations of the machine. The push-down stack of memory maps provided by the APEX time-sharing system is effective in some programming situations as a way to achieve a larger address space. In other situations, a more conventional overlay scheme is more satisfactory, and in still others, garbage collection techniques are indicated. The SPC is designed to support all these approaches to the address space problem but leaves to the programmer the decision of which techniques to use.

A first version of the SPC has been coded and is now approaching operational status. A first version of a speech labeling program is being revised to operate in the SPC environment. This program and a new speech segmentation program will serve to test the SPC during the next few months.

IV. TX-2 ACTIVITIES

A. BCPL

Work has continued on the development of the BCPL language and compliers operating on TX-2. BCPL is the language being used for system code in the TSP system and for development of the speech processing support system. Three BCPL compilers are currently operating on TX-2 generating code for TX-2, the SEL 810A test computers, and the BCOM machines in the TSP system.

The last Semiannual Technical Summary described a proposed modification to the BCPL compiler to facilitate manipulation of structured data. The necessary additions to the front end of the compiler are complete and have been debugged. The code generator changes for the TX-2 compiler are now ready for testing. The code generator changes for the BCOM version of the compiler will be undertaken during the next reporting period.

A new symbolic debugger for TX-2 BCPL programs is being designed. It will automatically relate each location in a BCPL binary code file with the corresponding program source text line. Also, it will relate the names of program variables (including subroutine argument names and locally defined variables) with their locations in memory. Accordingly, the BCPL compiler is being modified to put out information which links the output code to the source text, and to put out a symbol table. Also, a package of subroutines that implement an efficient hash-coded dictionary is being written.

The debugger will have facilities for setting traps in the program and on reference to data. Also, the user will be able to define and incrementally load subroutines into the debugger. These will be able to use the functions of the debugger, including functions which access the environment of the program. Such a subroutine can be associated with a specified trap to be executed whenever the trap occurs. For example, a subroutine could be used in this way to do conditional trapping or to gather statistics about program behavior. To facilitate this mode of operation, a new loader will be written to allow subroutines to be loaded as additions to existing loaded programs.

B. Gray Level Displays

In recent months, the use of the TX-2 graphics facility has expanded into areas that put different demands on the display system. In speech research work, the experimenter would like to manipulate speech parameters and rapidly view the results in relation to the original speech. The spectrogram, an amplitude-time-frequency distribution plot, is the most satisfactory representation of speech yet developed. In such a display, the magnitudes of the frequency components are presented by the intensity, or brightness, of areas of the spectrogram. Usually, spectrograms are produced by relatively slow hard-copying processes involving electrosensitive paper or photography. Conventional computer displays are ill suited to the display of spectrograms because such displays require large portions of the display area to be illuminated with different shadings, or gray levels. This can be achieved by a refresh type of CRT display, but large refresh memory and data rates are required, and flicker is often a problem. These considerations lead one to consider storage types of displays.

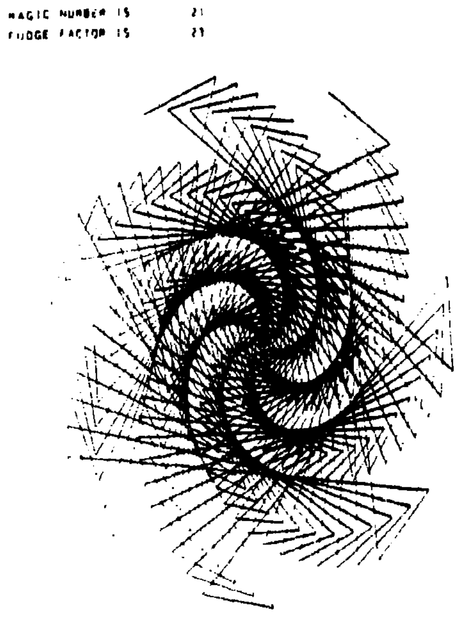
There are a number of Tektronix type 611 storage display units on TX-2, and they were tried for this application. They are bistable display units with no gray levels. Experiments using halftone techniques have shown that, while a gray level effect can be achieved, the resolution of the display suffers severely. Some other device is clearly needed.

During the past year, a number of relatively good and low-cost scan converters became available, and they promise to serve the need of displaying spectrograms. These units are electrical input to electrical output types of storage tubes. They accept various input formats and provide a TV raster type of output, and they have gray level capability. Scan converters have been available for some time, but the earlier units were very costly or had poor resolution.

A scan converter unit (the LCSC-1) was purchased from Hughes Aircraft Company. This is a self-contained unit, using a low-cost single-ended scan converter tube. It has a 50 percent modulation resolution of 1000 TV lines and some gray level capability. Experiments with this unit on the TX-2 computer show that the resolution is comparable to that of the Tektronix type 611 storage scope. About five intensity levels are discernible. Samples of the outputs are shown in Figs. 2 and 3. The scan converter tube used in this unit is an early engineering model and has some irregularity on its storage surface. We understand that current production tubes have improved characteristics, such as more shades of gray as well as better resolution. A new tube has been ordered and will be evaluated in the near future.

P91-471

Fig. 2. TX-2 generated gray levels as displayed by LCSC-1 scan converter.



```
MAGIC NUMBER IS        21
FUDGE FACTOR IS        21
```

P91-489

Fig. 3. TX-2 demonstration display showing alphanumeric and line drawing capability of LCSC-1 scan converter.

NOT REPRODUCIBLE